Convolutional Neural Networks for Food101 Dataset Report

Assignment 2 – Computer Vision

Thirada Tiamklang
14337188

94691 Deep Learning - Autumn 2024

1. Dataset Presentation

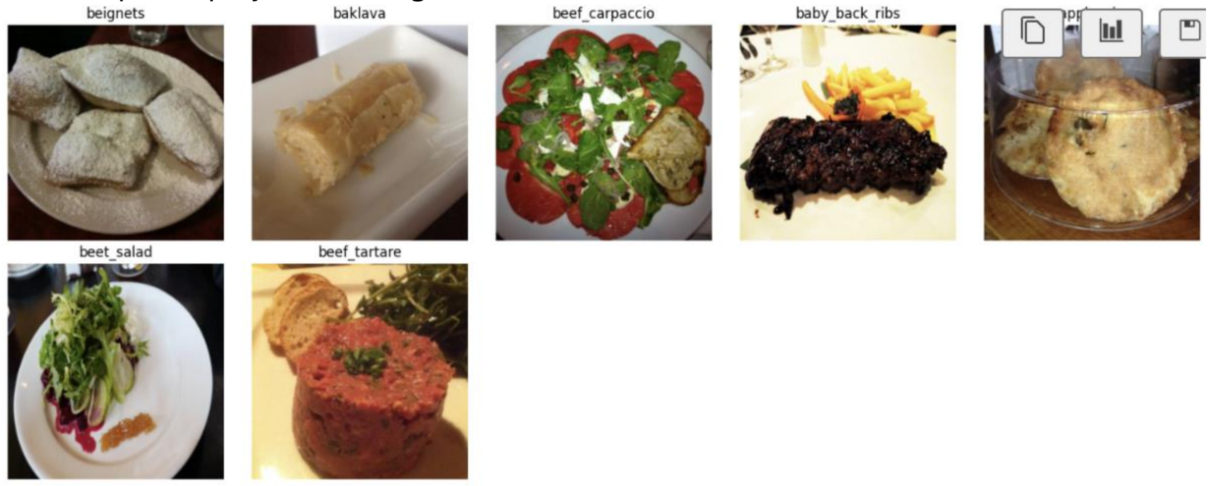
1.1 Dataset

The Food101 dataset comprises 101 food categories with a total of 101,000 images. Each class has 750 training images and 250 test images. All images are resized to have a maximum side length of 512 pixels.

1.2 Data Preparation

We loaded and transformed the dataset by resizing, converting to tensors, and normalizing the data. For each pre-trained model, we resized the images to different dimensions: 229x229 pixels for GoogLeNet, 224x224 pixels for MobileNet V3, and 331x331 pixels for NasNet. Due to GPU limitations, we used VS Code to run the notebook. We sampled 6 percent of the dataset or 6060 images and split it into training, validation, and test sets with a ratio of 0.7, 0.15, and 0.15, respectively. Therefore, the training set contains 4242 images, while both the validation and test sets contain 909 images each.

The sample display of the images shown below:



2. Architectural Analysis and Transfer Learning

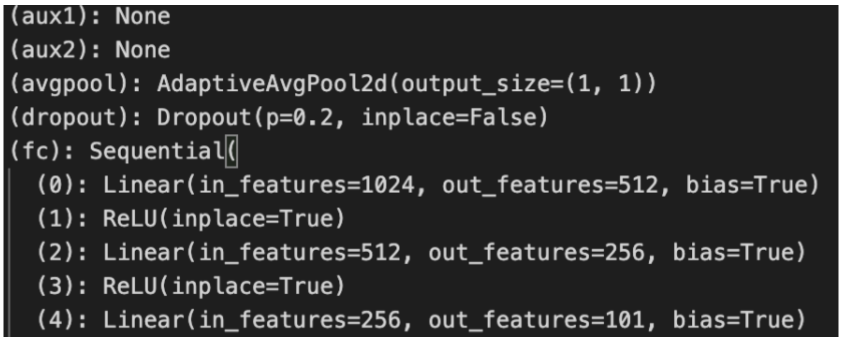
Three pre-trained CNN models are analyzed and compared for transfer learning to predict 101 different classes from the Food101 Dataset. For transfer learning, the pre-trained weights from the ImageNet dataset are utilized. The final classification head of each model is replaced with a global average pooling layer followed by three fully connected layers, with the last one serving as the output layer for predictions.

2.1 GoogLeNet

GoogLeNet known as Inception v1, is characterized by its deep architecture with inception modules. These modules allow for efficient computation by performing parallel convolutions of different sizes and concatenating their outputs. This architecture significantly reduces the number of parameters while maintaining high accuracy.

Images Transform:
GoogLeNet was originally trained on ImageNet with an input image size of 224x224 pixels. However, for this project, we used a slightly larger size of 229x229 pixels, which is also a common choice and can be suitable for GoogLeNet.

Model Architecture:
For this project, the pre-trained layers are frozen, and the classification head is replaced with a global average pooling layer followed by three fully connected layers with sizes 1024, 512, 256, and the output layer with 101 units for the 101 classes in the Food101 Dataset.



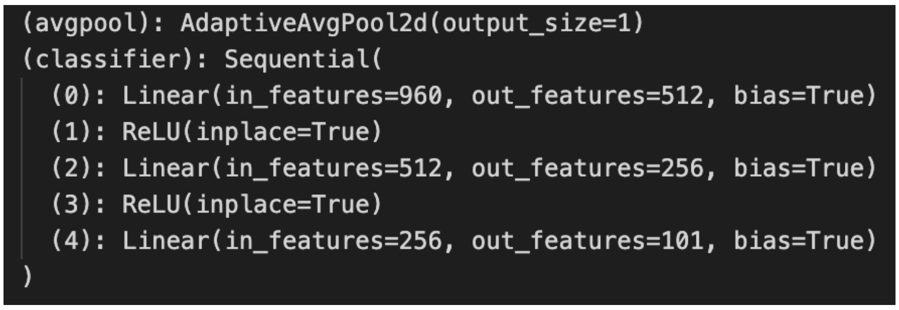
```
(aux1): None
(aux2): None
(avgpool): AdaptiveAvgPool2d(output_size=(1, 1))
(dropout): Dropout(p=0.2, inplace=False)
(fc): Sequential(
  (0): Linear(in_features=1024, out_features=512, bias=True)
  (1): ReLU(inplace=True)
  (2): Linear(in_features=512, out_features=256, bias=True)
  (3): ReLU(inplace=True)
  (4): Linear(in_features=256, out_features=101, bias=True)
```

Parameters:
- Criterion: CrossEntropy Loss
- Optimizer: Adam with learning rate = 0.00001
- LR Scheduler: Reduce LR on Plateau with mode='min', factor=0.1, patience=3, min_lr=1e-6

2.2 MobileNet V3
MobileNet V3 is designed to be lightweight and efficient, making it suitable for mobile and embedded applications. It introduces several architectural optimizations, including inverted residual blocks and linear bottlenecks. These optimizations reduce computational complexity while preserving performance.

Images Transform:
MobileNet V3 was designed for efficient mobile and embedded vision applications. It typically uses input image sizes of 224x224 pixels. This size is widely adopted and works well for most scenarios.

Model Architecture:
Similar to GoogLeNet, the pre-trained layers are frozen, and the classification head is replaced with a global average pooling layer followed by three fully connected layers with sizes 960, 512, 256, and the output layer with 101 units.

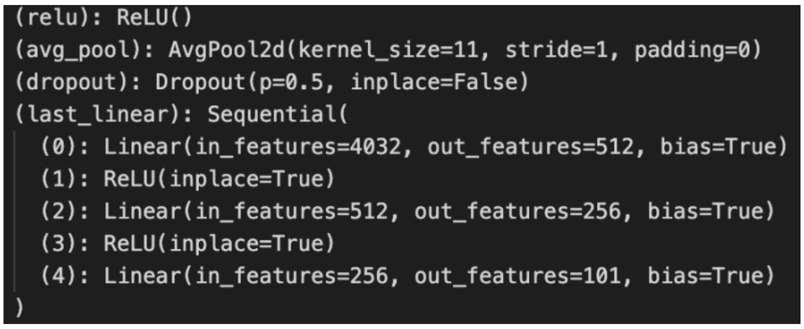

```
(avgpool): AdaptiveAvgPool2d(output_size=1)
(classifier): Sequential(
  (0): Linear(in_features=960, out_features=512, bias=True)
  (1): ReLU(inplace=True)
  (2): Linear(in_features=512, out_features=256, bias=True)
  (3): ReLU(inplace=True)
  (4): Linear(in_features=256, out_features=101, bias=True)
)
```

Parameters:
- Criterion: CrossEntropy Loss
- Optimizer: Adam with learning rate = 0.00001
- LR Scheduler: Reduce LR on Plateau with mode='min', factor=0.1, patience=3, min_lr=1e-6

2.3 NASNet
NASNet, short for Neural Architecture Search Network, is the result of a neural architecture search process. This process explores a vast search space of possible architectures to find optimal models automatically. NASNet architectures are highly diverse and often incorporate novel architectural elements discovered during the search process.

Images Transform:
NasNet was trained on ImageNet with an input size of 331x331 pixels. This size allows for a more detailed representation of images, capturing finer features compared to smaller input sizes.

Model Architecture:
The pre-trained layers are frozen, and the classification head is replaced with a global average pooling layer followed by three fully connected layers with sizes 4032, 512, 256, and the output layer with 101 units.


```
(relu): ReLU()
(avg_pool): AvgPool2d(kernel_size=11, stride=1, padding=0)
(dropout): Dropout(p=0.5, inplace=False)
(last_linear): Sequential(
  (0): Linear(in_features=4032, out_features=512, bias=True)
  (1): ReLU(inplace=True)
  (2): Linear(in_features=512, out_features=256, bias=True)
  (3): ReLU(inplace=True)
  (4): Linear(in_features=256, out_features=101, bias=True)
)
```

Parameters:
- Criterion: CrossEntropy Loss
- Optimizer: Adam with learning rate = 0.00001
- LR Scheduler: Reduce LR on Plateau with mode='min', factor=0.1, patience=3, min_lr=1e-6

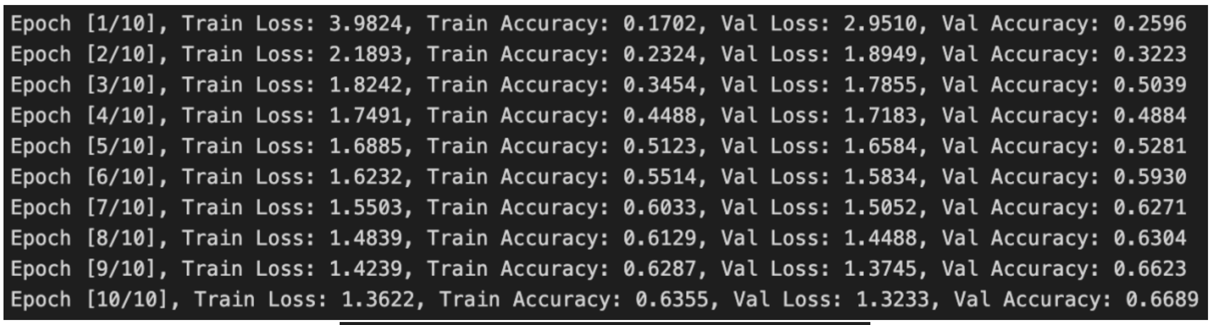
3. Training Process and Results

The transfer learning training process involves fine-tuning the pre-trained models on the Food101 dataset. The models are trained using techniques such as data augmentation, batch normalization, and learning rate scheduling.

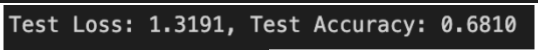
The performance of each model is evaluated in terms of accuracy on the test set. Additionally, the computational efficiency and memory footprint of each model are considered.

3.1 GoogLeNet

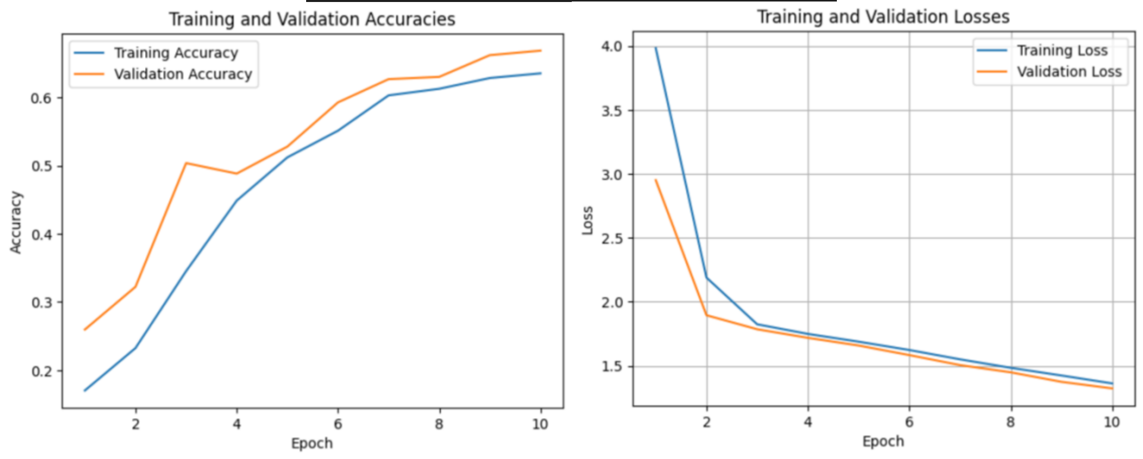
We trained the GoogLeNet model for 10 epochs, and the results of accuracy and loss for both the training and validation sets are shown below:



```
Epoch [1/10], Train Loss: 3.9824, Train Accuracy: 0.1702, Val Loss: 2.9510, Val Accuracy: 0.2596
Epoch [2/10], Train Loss: 2.1893, Train Accuracy: 0.2324, Val Loss: 1.8949, Val Accuracy: 0.3223
Epoch [3/10], Train Loss: 1.8242, Train Accuracy: 0.3454, Val Loss: 1.7855, Val Accuracy: 0.5039
Epoch [4/10], Train Loss: 1.7491, Train Accuracy: 0.4488, Val Loss: 1.7183, Val Accuracy: 0.4884
Epoch [5/10], Train Loss: 1.6885, Train Accuracy: 0.5123, Val Loss: 1.6584, Val Accuracy: 0.5281
Epoch [6/10], Train Loss: 1.6232, Train Accuracy: 0.5514, Val Loss: 1.5834, Val Accuracy: 0.5930
Epoch [7/10], Train Loss: 1.5503, Train Accuracy: 0.6033, Val Loss: 1.5052, Val Accuracy: 0.6271
Epoch [8/10], Train Loss: 1.4839, Train Accuracy: 0.6129, Val Loss: 1.4488, Val Accuracy: 0.6304
Epoch [9/10], Train Loss: 1.4239, Train Accuracy: 0.6287, Val Loss: 1.3745, Val Accuracy: 0.6623
Epoch [10/10], Train Loss: 1.3622, Train Accuracy: 0.6355, Val Loss: 1.3233, Val Accuracy: 0.6689
```



```
Test Loss: 1.3191, Test Accuracy: 0.6810
```



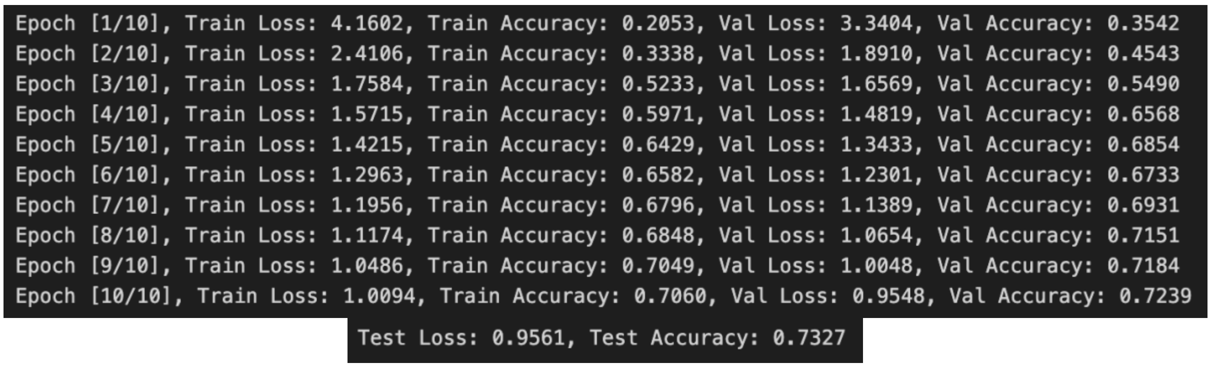
The training of GoogLeNet on the Food101 dataset exhibited a progressive decrease in the loss metric as the model iterated through epochs. Initially, the model encountered a relatively high training loss of 3.9824, with a corresponding training accuracy of 0.1702 in the first epoch. However, as the training progressed, both the training loss and training accuracy improved steadily across subsequent epochs.

By the final epoch, the training loss had significantly decreased to 1.3622, accompanied by a corresponding training accuracy of 0.6355. This downward trend in training loss indicates that the model's parameter adjustments effectively minimized the error between predicted and actual values during training.

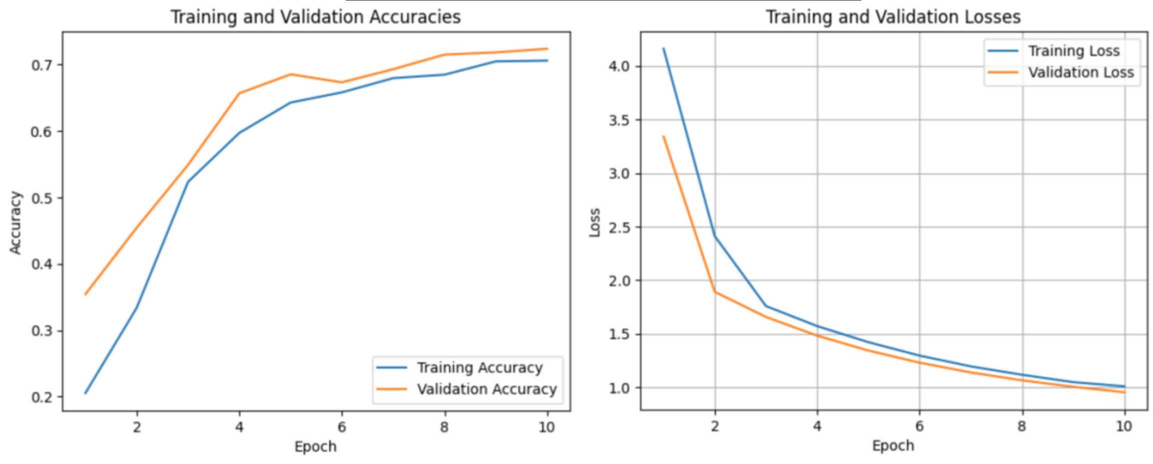
Concurrently, the validation loss also exhibited a decreasing trend throughout the epochs, indicating that the model's generalization performance improved over time. Similarly, the validation accuracy experienced a gradual increase, reflecting the model's enhanced ability to classify unseen data accurately.

Ultimately, the model's performance was evaluated on the test set, where it achieved a test loss of 1.3191 and a test accuracy of 0.6810. These metrics indicate that the model's learned parameters generalized well to unseen data, achieving a reasonable level of accuracy in classifying food images from the Food101 dataset.

3.2 MobileNet V3
We trained the MobileNet V3model for 10 epochs, and the results of accuracy and loss for both the training and validation sets are shown below:



```
Epoch [1/10], Train Loss: 4.1602, Train Accuracy: 0.2053, Val Loss: 3.3404, Val Accuracy: 0.3542
Epoch [2/10], Train Loss: 2.4106, Train Accuracy: 0.3338, Val Loss: 1.8910, Val Accuracy: 0.4543
Epoch [3/10], Train Loss: 1.7584, Train Accuracy: 0.5233, Val Loss: 1.6569, Val Accuracy: 0.5490
Epoch [4/10], Train Loss: 1.5715, Train Accuracy: 0.5971, Val Loss: 1.4819, Val Accuracy: 0.6568
Epoch [5/10], Train Loss: 1.4215, Train Accuracy: 0.6429, Val Loss: 1.3433, Val Accuracy: 0.6854
Epoch [6/10], Train Loss: 1.2963, Train Accuracy: 0.6582, Val Loss: 1.2301, Val Accuracy: 0.6733
Epoch [7/10], Train Loss: 1.1956, Train Accuracy: 0.6796, Val Loss: 1.1389, Val Accuracy: 0.6931
Epoch [8/10], Train Loss: 1.1174, Train Accuracy: 0.6848, Val Loss: 1.0654, Val Accuracy: 0.7151
Epoch [9/10], Train Loss: 1.0486, Train Accuracy: 0.7049, Val Loss: 1.0048, Val Accuracy: 0.7184
Epoch [10/10], Train Loss: 1.0094, Train Accuracy: 0.7060, Val Loss: 0.9548, Val Accuracy: 0.7239
Test Loss: 0.9561, Test Accuracy: 0.7327
```



During the training phase, the MobileNet V3 model was trained for 10 epochs. At the outset of training, the model exhibited a relatively high training loss of 4.1602, accompanied by a training accuracy of 0.2053. However, as the training progressed, both the training loss and accuracy improved consistently.

By the conclusion of the training process, the model achieved a training loss of 1.0094 and a training accuracy of 0.7060. This indicates that the model successfully learned to classify the food images in the training set with a high degree of accuracy.

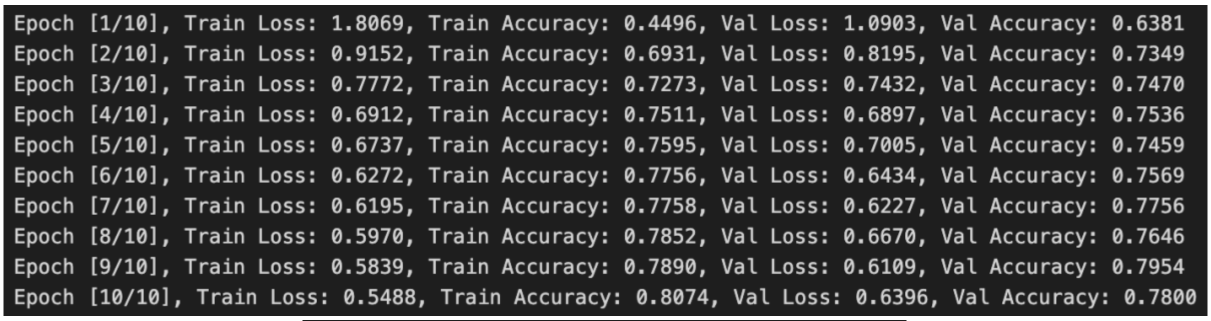
Similarly, the validation loss decreased from 3.3404 to 0.9548, with the validation accuracy increasing from 0.3542 to 0.7239. This suggests that the model was able to generalize well to unseen data, as evidenced by its performance on the validation set.

Upon evaluating the model on the test set, it achieved a test loss of 0.9561 and a test accuracy of 0.7327. This further confirms the generalization capability of the model, as it performed well on previously unseen data.

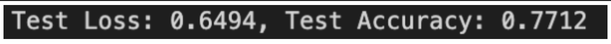
Overall, the training process demonstrates the effectiveness of the MobileNet V3 model in learning the patterns present in the Food101 dataset, as evidenced by the consistent decrease in loss and increase in accuracy over the course of training, as well as its robust performance on the test set.

3.3 NASNet

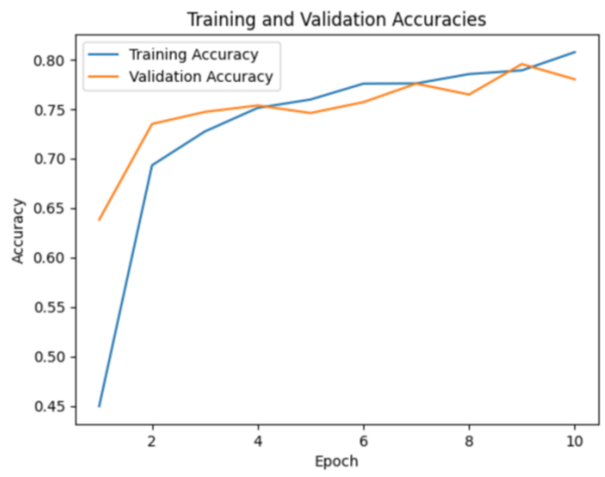
We trained the NASNet model for 10 epochs, and the results of accuracy and loss for both the training and validation sets are shown below:

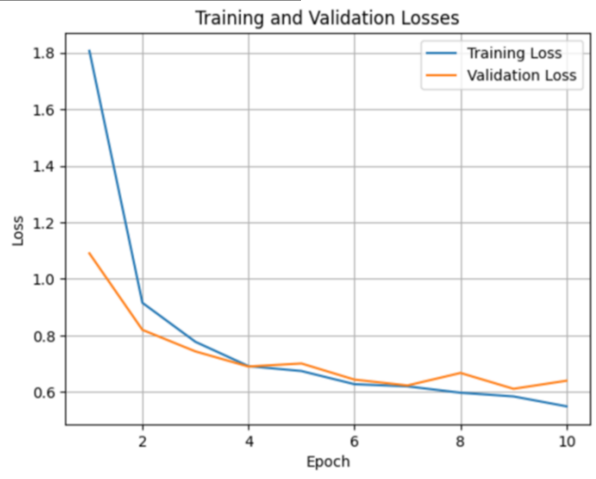


```
Epoch [1/10], Train Loss: 1.8069, Train Accuracy: 0.4496, Val Loss: 1.0903, Val Accuracy: 0.6381
Epoch [2/10], Train Loss: 0.9152, Train Accuracy: 0.6931, Val Loss: 0.8195, Val Accuracy: 0.7349
Epoch [3/10], Train Loss: 0.7772, Train Accuracy: 0.7273, Val Loss: 0.7432, Val Accuracy: 0.7470
Epoch [4/10], Train Loss: 0.6912, Train Accuracy: 0.7511, Val Loss: 0.6897, Val Accuracy: 0.7536
Epoch [5/10], Train Loss: 0.6737, Train Accuracy: 0.7595, Val Loss: 0.7005, Val Accuracy: 0.7459
Epoch [6/10], Train Loss: 0.6272, Train Accuracy: 0.7756, Val Loss: 0.6434, Val Accuracy: 0.7569
Epoch [7/10], Train Loss: 0.6195, Train Accuracy: 0.7758, Val Loss: 0.6227, Val Accuracy: 0.7756
Epoch [8/10], Train Loss: 0.5970, Train Accuracy: 0.7852, Val Loss: 0.6670, Val Accuracy: 0.7646
Epoch [9/10], Train Loss: 0.5839, Train Accuracy: 0.7890, Val Loss: 0.6109, Val Accuracy: 0.7954
Epoch [10/10], Train Loss: 0.5488, Train Accuracy: 0.8074, Val Loss: 0.6396, Val Accuracy: 0.7800
```



```
Test Loss: 0.6494, Test Accuracy: 0.7712
```





Throughout the training process, the NASNet model was trained for 10 epochs. Initially, the training loss was relatively high at 1.8069, accompanied by a corresponding training accuracy of 0.4496. However, as training progressed, both the training loss and validation loss exhibited a decreasing trend, indicating an improvement in the model's performance.

By the end of the training, the training loss decreased to 0.5488, while the training accuracy increased to 0.8074. Similarly, the validation loss decreased to 0.6396, with a corresponding validation accuracy of 0.7800. This suggests that the model was able to generalize well to unseen data, as evidenced by the relatively high validation accuracy.

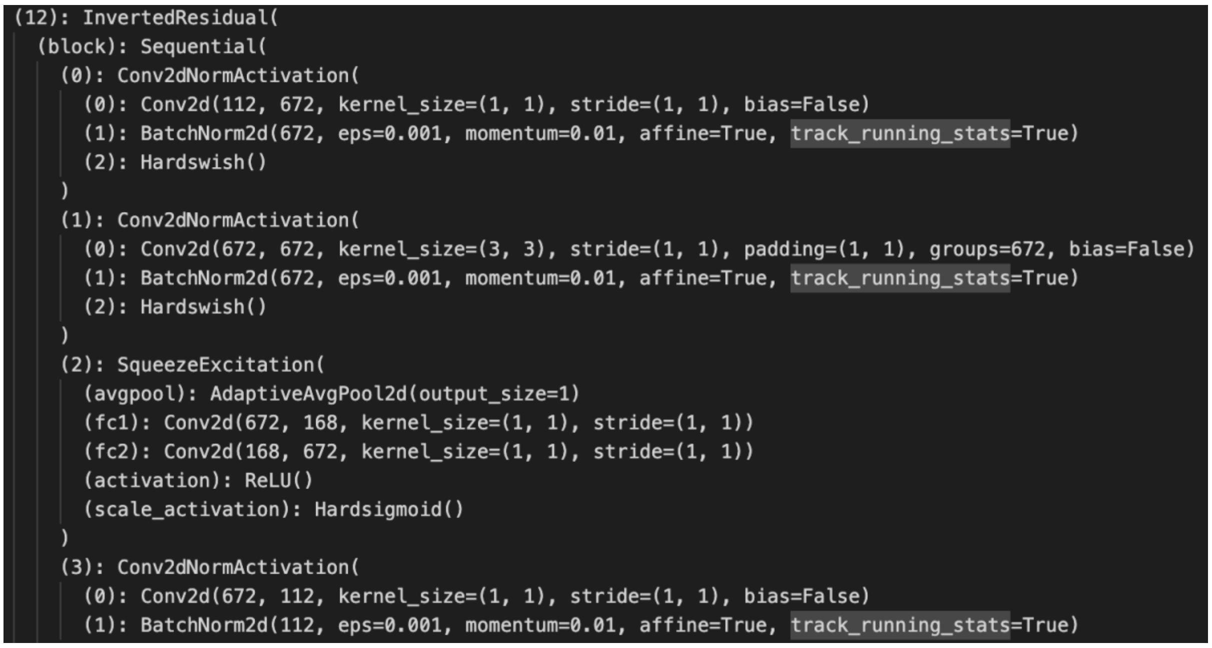
Upon evaluating the model on the test set, the test loss was found to be 0.6494, with a test accuracy of 0.7712. This further confirms the generalization capability of the model, as it performed well on previously unseen data.

Overall, the training process demonstrates the effectiveness of the NASNet model in learning the patterns present in the Food101 dataset, as evidenced by the consistent decrease in loss and increase in accuracy over the course of training, as well as its robust performance on the test set.

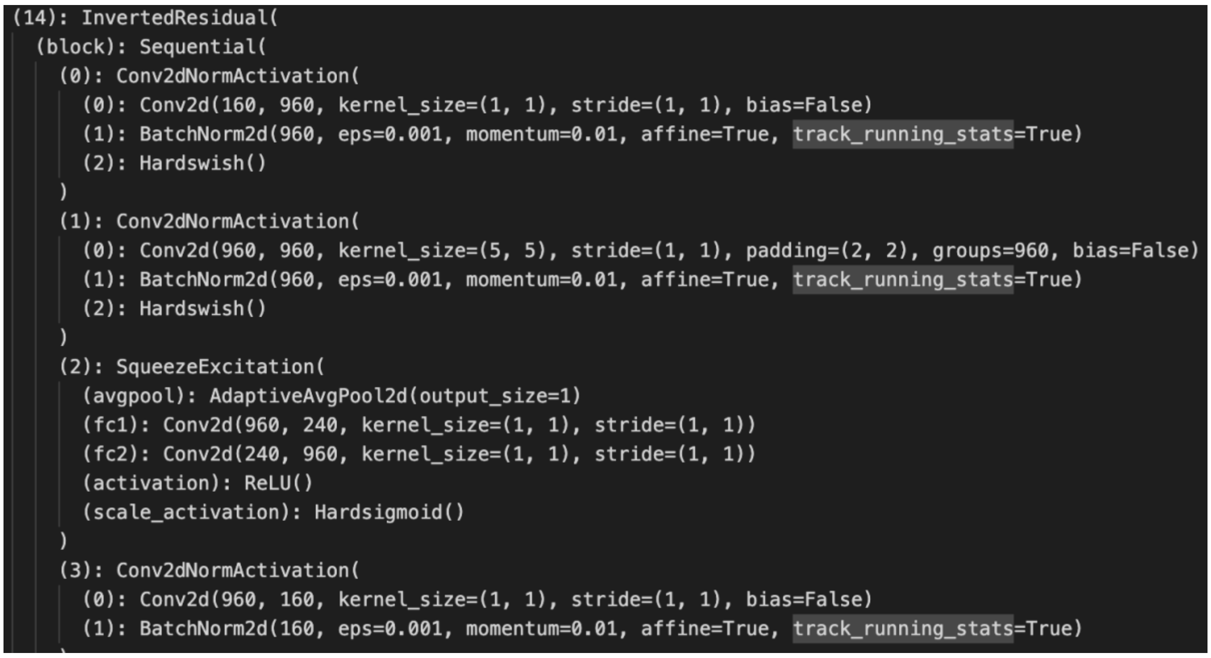
4. Model Selection and Fine-Tuning

Based on the performance from Part A, the best-performing pre-trained model is NASNet with the accuracy on test set 0.7712 followed by MobileNet V3 with 0.7327. However, due to the lengthy GPU runtime and limited computational resources, there are constraints on the fine-tuning process and dataset size with NASNet model. NASNet's intricate architecture involves complex operations such as cell search and stacking, which may require more data and computational resources to fine-tune effectively. On the other hand, MobileNetV3's streamlined architecture allows for faster convergence and effective utilization of limited data. Therefore, we will be fine-tuning the MobileNet V3 instead.

The fine-tuning process involves unfreezing specific layers of the model and re-training them with a lower learning rate. With two different layers are experimented with for unfreezing to observe their impact on model performance and convergence speed. In our approach, we unfreeze the specified layers on layer 12 and 14 which are:



```
(12): InvertedResidual(
  (block): Sequential(
    (0): Conv2dNormActivation(
      (0): Conv2d(112, 672, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (1): BatchNorm2d(672, eps=0.001, momentum=0.01, affine=True, track_running_stats=True)
      (2): Hardswish()
    )
    (1): Conv2dNormActivation(
      (0): Conv2d(672, 672, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=672, bias=False)
      (1): BatchNorm2d(672, eps=0.001, momentum=0.01, affine=True, track_running_stats=True)
      (2): Hardswish()
    )
    (2): SqueezeExcitation(
      (avgpool): AdaptiveAvgPool2d(output_size=1)
      (fc1): Conv2d(672, 168, kernel_size=(1, 1), stride=(1, 1))
      (fc2): Conv2d(168, 672, kernel_size=(1, 1), stride=(1, 1))
      (activation): ReLU()
      (scale_activation): Hardsigmoid()
    )
    (3): Conv2dNormActivation(
      (0): Conv2d(672, 112, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (1): BatchNorm2d(112, eps=0.001, momentum=0.01, affine=True, track_running_stats=True)
```

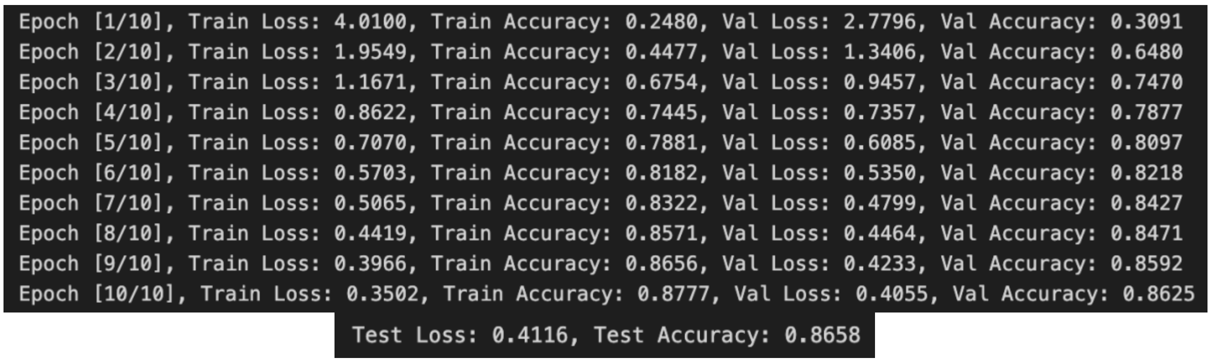


```
(14): InvertedResidual(
  (block): Sequential(
    (0): Conv2dNormActivation(
      (0): Conv2d(160, 960, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (1): BatchNorm2d(960, eps=0.001, momentum=0.01, affine=True, track_running_stats=True)
      (2): Hardswish()
    )
    (1): Conv2dNormActivation(
      (0): Conv2d(960, 960, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2), groups=960, bias=False)
      (1): BatchNorm2d(960, eps=0.001, momentum=0.01, affine=True, track_running_stats=True)
      (2): Hardswish()
    )
    (2): SqueezeExcitation(
      (avgpool): AdaptiveAvgPool2d(output_size=1)
      (fc1): Conv2d(960, 240, kernel_size=(1, 1), stride=(1, 1))
      (fc2): Conv2d(240, 960, kernel_size=(1, 1), stride=(1, 1))
      (activation): ReLU()
      (scale_activation): Hardsigmoid()
    )
    (3): Conv2dNormActivation(
      (0): Conv2d(960, 160, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (1): BatchNorm2d(160, eps=0.001, momentum=0.01, affine=True, track_running_stats=True)
```

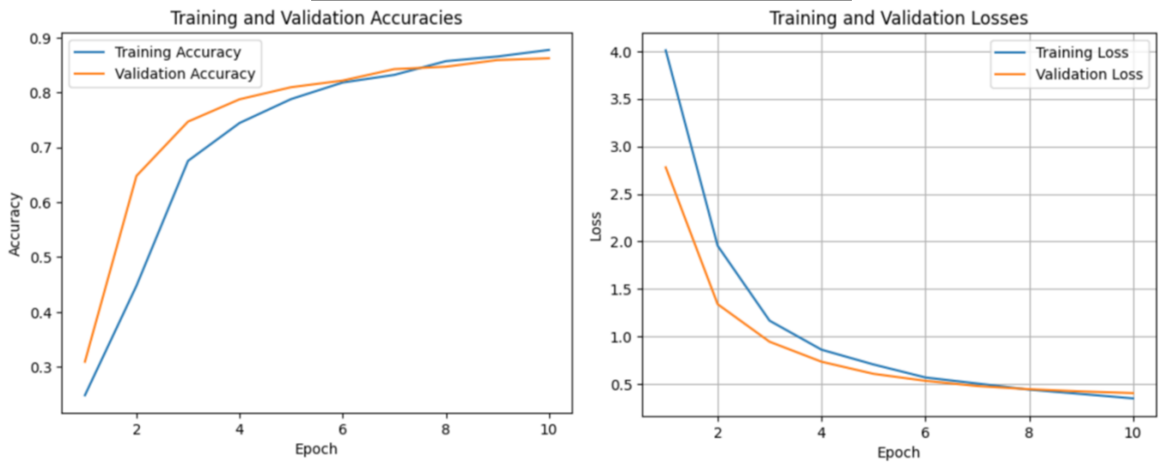
5. Performance Analysis and Recommendations

5.1 Performance Analysis of MobileNetV3

The fine-tuned MobileNetV3 model demonstrates notable improvements in performance over the training epochs, as evidenced by both training and validation accuracy.



```
Epoch [1/10], Train Loss: 4.0100, Train Accuracy: 0.2480, Val Loss: 2.7796, Val Accuracy: 0.3091
Epoch [2/10], Train Loss: 1.9549, Train Accuracy: 0.4477, Val Loss: 1.3406, Val Accuracy: 0.6480
Epoch [3/10], Train Loss: 1.1671, Train Accuracy: 0.6754, Val Loss: 0.9457, Val Accuracy: 0.7470
Epoch [4/10], Train Loss: 0.8622, Train Accuracy: 0.7445, Val Loss: 0.7357, Val Accuracy: 0.7877
Epoch [5/10], Train Loss: 0.7070, Train Accuracy: 0.7881, Val Loss: 0.6085, Val Accuracy: 0.8097
Epoch [6/10], Train Loss: 0.5703, Train Accuracy: 0.8182, Val Loss: 0.5350, Val Accuracy: 0.8218
Epoch [7/10], Train Loss: 0.5065, Train Accuracy: 0.8322, Val Loss: 0.4799, Val Accuracy: 0.8427
Epoch [8/10], Train Loss: 0.4419, Train Accuracy: 0.8571, Val Loss: 0.4464, Val Accuracy: 0.8471
Epoch [9/10], Train Loss: 0.3966, Train Accuracy: 0.8656, Val Loss: 0.4233, Val Accuracy: 0.8592
Epoch [10/10], Train Loss: 0.3502, Train Accuracy: 0.8777, Val Loss: 0.4055, Val Accuracy: 0.8625
Test Loss: 0.4116, Test Accuracy: 0.8658
```



- **Epoch 1:** The training accuracy starts at 24.80% and validation accuracy at 30.91%, indicating a relatively low initial performance.
- **Epoch 2:** Significant improvement is observed, with the training accuracy rising to 44.77% and validation accuracy to 64.80%.
- **Epoch 3:** Both training and validation accuracies continue to increase substantially, reaching 67.54% and 74.70%, respectively.
- **Epochs 4-10:** The model continues to refine its performance, with both training and validation accuracies showing consistent growth. By the final epoch, the training accuracy reaches 87.77%, while the validation accuracy reaches 86.25%.

These results indicate that the model effectively learns from the training data and generalizes well to unseen validation data. The decreasing trend in loss values across epochs further confirms the model's convergence and ability to minimize prediction errors.

Comparing the results with those from Part A, the fine-tuned MobileNetV3 model shows a substantial improvement in test set accuracy, achieving 86.58%. This represents a significant improvement of 13% over the accuracy achieved in Part A. It underscores the effectiveness of fine-tuning specific layers of MobileNetV3, demonstrating its capability to adapt to the food image classification task.

Overall, the results highlight the efficacy of fine-tuning MobileNetV3 by unfreezing specific layers, leading to notable enhancements in accuracy on the test set. This approach showcases the potential of transfer learning for improving model performance in food image classification tasks.

5.2 Recommendations and Remaining Issues
While the fine-tuned MobileNetV3 model demonstrates promising performance, several recommendations and remaining issues can be addressed to further enhance its effectiveness:

1. Evaluation on Test Set: Although the model performs well on the validation set, it's essential to evaluate its performance on a separate test set to ensure unbiased performance assessment.
2. Analysis of Overfitting: While the model exhibits significant improvement in accuracy, monitoring for signs of overfitting, such as a large gap between training and validation accuracies, should be continued. Regularization techniques like dropout or data augmentation could be explored to mitigate overfitting if observed.
3. Fine-Tuning Hyperparameters: Experimenting with different hyperparameters, such as learning rate schedules, optimizer choices, and batch sizes, may further optimize the model's performance and convergence speed.
4. Utilization of Larger Dataset: Training the model on a larger portion of the dataset or utilizing data augmentation techniques can provide more diverse examples for the model to learn from, potentially leading to better generalization.

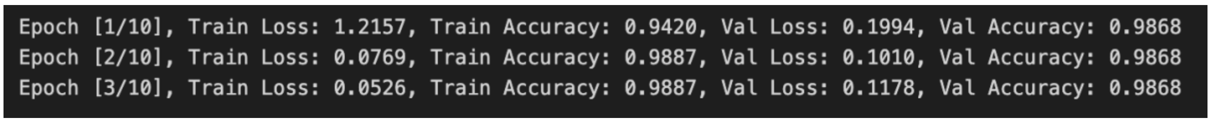
5. Exploration of Model Architectures: Considering the rapid advancements in deep learning, exploring alternative model architectures or assembling multiple models could yield further improvements in performance.

In conclusion, while the fine-tuned MobileNetV3 model demonstrates commendable performance, continued experimentation and optimization efforts are necessary to unlock its full potential for food image classification tasks.

6. Limitations

The limitations encountered during this phase highlight the constraints imposed by GPU runtime, which hindered the fine-tuning of NasNet. The lengthy runtime required for training complex models like NasNet on local hardware indicates the need for more powerful computational resources, such as cloud platforms. However, the cost implications associated with using cloud platforms must also be considered.

Despite efforts to fine-tune NasNet by unfreezing specific layers, such as comb_iter_4_left and comb_iter_3_left, using a reduced dataset size of 1 percent, the process was still computationally intensive. For instance, training for just three epochs consumed approximately 300 minutes before the kernel died, indicating significant challenges in achieving convergence within reasonable timeframes.



```
Epoch [1/10], Train Loss: 1.2157, Train Accuracy: 0.9420, Val Loss: 0.1994, Val Accuracy: 0.9868
Epoch [2/10], Train Loss: 0.0769, Train Accuracy: 0.9887, Val Loss: 0.1010, Val Accuracy: 0.9868
Epoch [3/10], Train Loss: 0.0526, Train Accuracy: 0.9887, Val Loss: 0.1178, Val Accuracy: 0.9868
```

In conclusion, the analysis and experimentation with pre-trained CNN models on the Food101 dataset provide valuable insights into their effectiveness for transfer learning and fine-tuning tasks. However, overcoming challenges related to computational resources and runtime constraints remains crucial for optimizing model performance. Leveraging cloud platforms for training could offer a viable solution, albeit with considerations for cost-effectiveness. Further exploration of optimization techniques and model architectures may also yield improvements in convergence speed and overall performance.